Page F Node Numbers

Network diagnostics addition

Wed, Dec 30, 1998

When the Network Frames page (PAGENETF) displays frames that use IP, it indicates the node number by using a pseudo node#, which corresponds to a socket, in that it includes (via the IPARP table) the IP address and the UDP port#. But for viewing these diagnostics, this is often somewhat obscure, as the meaning of pseudo node numbers depends upon how the IPARP table gets filled. This note discusses how to show the node# more meaningfully, in many cases.

In actual practice, the port#s we use are not so random. All Classic protocol messages use port# 6800, and all Acnet-based messages use port #6801. The idea here is to show the node# with one of two indicator symbols when the port# is actually one of these two cases—very common cases at Fermilab. For example, if a pseudo node# `E061` translates to the IP address of `node06b3`, then we might show it as `.6B3` in the case of Classic protocol, or `:6B3` in the case of Acnet protocols. If neithor of these two special port#s is referenced, we must leave it displayed as the pseudo node#.

The problem of working out these cases means that a separate data request must be made to the target node (from which network diagnostics have been collected for display) to get the information needed to effect the translation. A new listype (87) has been added to the system code to support access to IPARP table entries. Its ident has the form of a channel# ident, but the index value is an IPARP table index#. One can derive the IPARP table index# easily from the pseudo node#. It is simply the middle 8 bits of the word. For example, pseudo node# `E061` refers to table index 6.

The structure returned from access to an IPARP table entry is 16 bytes long, as follows:

physical address	6 bytes
node#	2 bytes
IP address	4 bytes
ptr to Port# block	4 bytes

The Port# block has the following structure:

```
MBLK21   RECORD 0
MBLKSIZE DS     1       ;Size of memory block
PORTICNT DS     1       ;Input activity counter
PORTOCNT DS     1       ;Output activity counter
MBLKTYPE DS     1       ;Memory block type#
NXPORT   DS     1       ;Offset to next port# word to try
NPORTS   DS     1       ;#ports active
PTIMOUT  DS     1       ;Seconds timeout counter
ARPTIMO  DS     1       ;ARP re-send timeout counter
DTGRAMI  DS.L   1       ;Ptr to linked list of incomplete datagrams
DTGRAMS  DS.L   1       ;Ptr to linked list of complete datagrams
ARPQUEUE DS.L   1       ;Ptr to queued datagrams awaiting ARP reply
         DS.L   1       ;spare
PORTS    DS     MXPORTS*2 ;Array of active source port#s, use counts
MBLK21S  EQU    *
         ENDR
```

The interesting part is the array of Port#s at offset 32 bytes into this structure. Each element of this 16-element array consists of a 2-byte port# and a 2-byte use count. To derive the port# index into this array from the pseudo node#, merely use the least significant 4 bits. For example, for pseudo node# `E061`, the port# array index is 1. The index values range from 0–15. Element 0 of this array always has the port# word set to 0, allowing for up to 15 nonzero port#s associated with each IP address to be active at one time.

In order to determine the node# that corresponds to the IP address obtained from the IPARP table entry, one must use listype 81, which supports access to IPNAT entries. Each entry in this table is 8 bytes in length, as follows:

node#	2 bytes
counters	2 bytes
IP address	4 bytes

The order of entries in this table is usually increasing, but new entruies all always added at the end. This table is updated with information obtained from the Domain Name Server, via the LOOPDNSQ local application. The table from any node may be used, but some nodes may include more than others. It is the initiated request from that node which fill the table. At reset time, the table entries are sorted, but are not cleared. If one cannot find a match in this table, one may use the node# that is found in the IPARP table entry instead.

From the above descriptions, one should be able to devise a scheme for being able to translate the pseudo node#s into the above suggested format. This logic would need to be added to the PAGENETF program.

After collecting the data that is to be displayed, scan through all the node numbers, and build a list of all unique pseudo node numbers. Build a list of IPARP table index numbers that correspond to these pseudo node numbers. Request the 16 bytes of data for each using listype 87. From the port# block pointers, build a memory request for 64 bytes of port number arrays. Read out the IPNAT from the target node, or from a specific node that has a fairly full IPNAT. This should prvide what is needed to modify the listing of frames, so that for IP-based datagrams, it could indicate the real node number and whether the Classic or Acnet port was used.